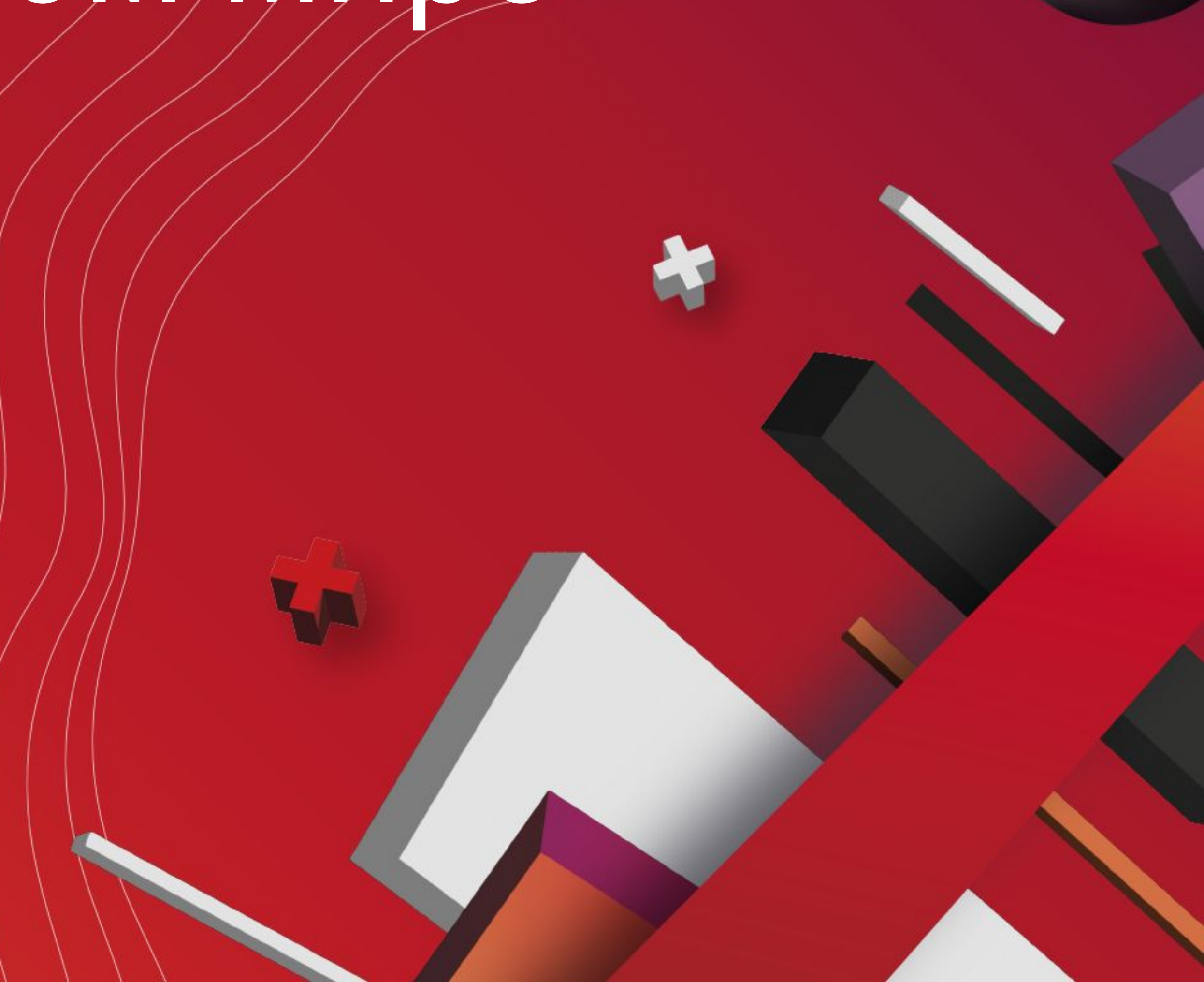
PHP в асинхронном мире

МИР АСИНХРОНЕН

Мамонтов Дмитрий
Руководитель команды MAGNIT





«Почему, — сказал Додо, — лучший способ объяснить это — сделать это». (И, если вы захотите попробовать это сами, в один из зимних дней я расскажу вам, как это удалось Додо.) — Льюис Кэрролл, Алиса в стране чудес



О чём будем говорить. Основы

- Асинхронность в реальном мире



О чём будем говорить. Основы

- Асинхронность в реальном мире
- Как устроен PHP



О чём будем говорить. Основы

- Асинхронность в реальном мире
- Как устроен PHP
- Реальный пример?



О чём будем говорить. Fiber’ы

Что это такое?



О чём будем говорить. Fiberʼы

- Что это такое?
- Принесли ли Fiberʼы асинхронность в PHP?



О чём будем говорить. Fiber’ы

- Что это такое?
- Принесли ли Fiber’ы асинхронность в PHP?
- Нужны ли нам Fiber’ы?



О чём будем говорить. Fiber’ы

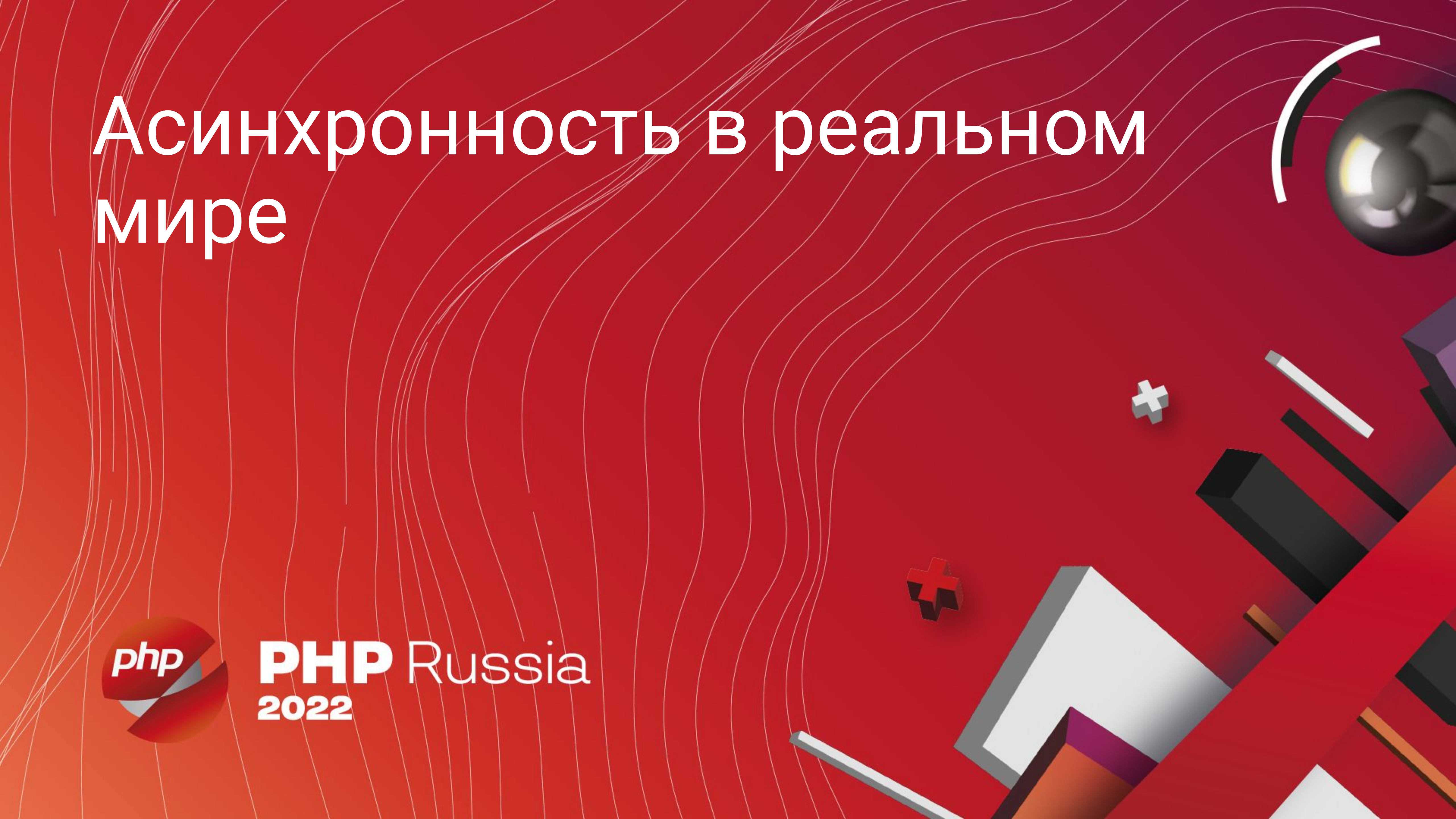
- Что это такое?
- Принесли ли Fiber’ы асинхронность в PHP?
- Нужны ли нам Fiber’ы?
- Какую проблему решают Fiber’ы?



О чём будем говорить. Fiber’ы

- Что это такое?
- Принесли ли Fiber’ы асинхронность в PHP?
- Нужны ли нам Fiber’ы?
- Какую проблему решают Fiber’ы?
- Все глубже и глубже




Асинхронность в реальном мире
php
PHP Russia
2022


php
PHP Russia
2022

LUNGS

HEART

LARYNX & THYROID

KINDEYS

PANCREAS

LIVER

STOMACH

INTESTINES

BRAIN

SPLEEN



«А это чудеса» – равнодушно пояснил Чеширский Кот.



Как устроен PHP

ВНИЗ ПО КРОЛИЧЬЕЙ НОРЕ

PHP Russia
2022

Запуск приложения

- Встроенный веб-сервер



Запуск приложения

- Встроенный веб-сервер
- Интерфейс командной строки



Запуск приложения

- Встроенный веб-сервер
- Интерфейс командной строки
- Fast Process Manager



Интерфейс командной строки





Fast Process Manager (FPM)

SOCKET

PHP-FPM → PHP → Application

PHP-FPM → PHP → Application

PHP-FPM → PHP → Application

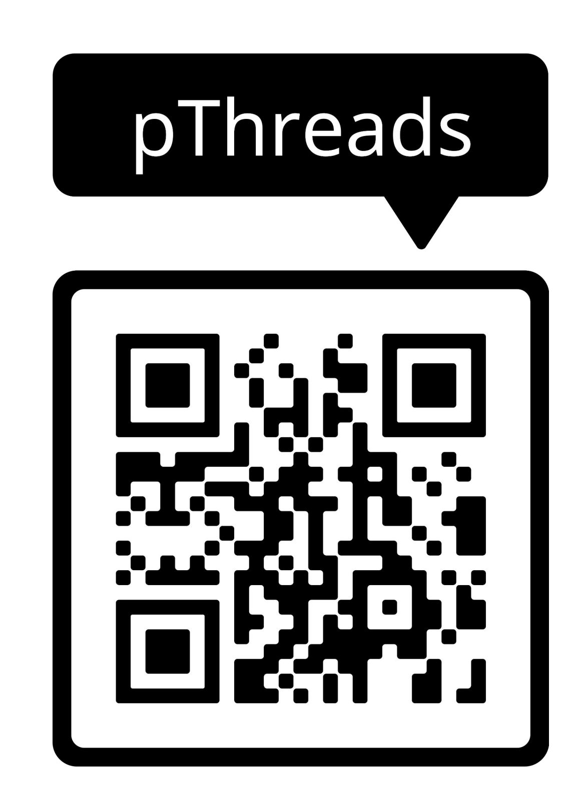


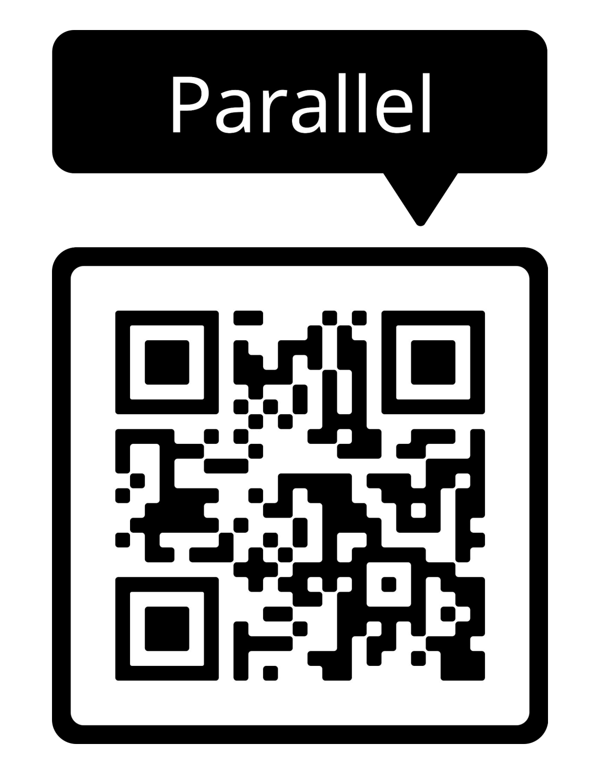
Как устроен PHP

Вы уже отгадали загадку?



Zend Thread Safety







Start Task D *Result*

Start Task A *Result*

Time

Start Task C *Result*

Start Task B *Result*



Start Task A Result
Start Task B Result
Start Task C Result
Start Task D Result
Time

Queue



Start Task D Result

Start Task A Result

Time

Start Task C Result

Start Task B Result




Лучший способ объяснить -
это запрограммировать
php
PHP Russia
2022

Promise
Resolve
Promise
Resolve

Start
Task A
Waiting...
Task A
Result
Time
Start
Task B
Waiting...
Task B
Result

External Request
Reference
External Request
Reference
Check by Reference
Result
Check by Reference
Result

Data Store Client



```php
<?php

final class Promise implements PromiseInterface
{
    public const STATE_PENDING = "pending";
    public const STATE_FULFILLED = "fulfilled";
    public const STATE_REJECTED = "rejected";

    public function __construct()
    {
        $this->state = self::STATE_PENDING;
    }

    public function resolve(mixed $result): void
    {
        $this->state = self::STATE_FULFILLED;
        $this->result = $result;
    }

    public function reject(): void
    {
        $this->state = self::STATE_REJECTED;
        unset($this->result);
    }

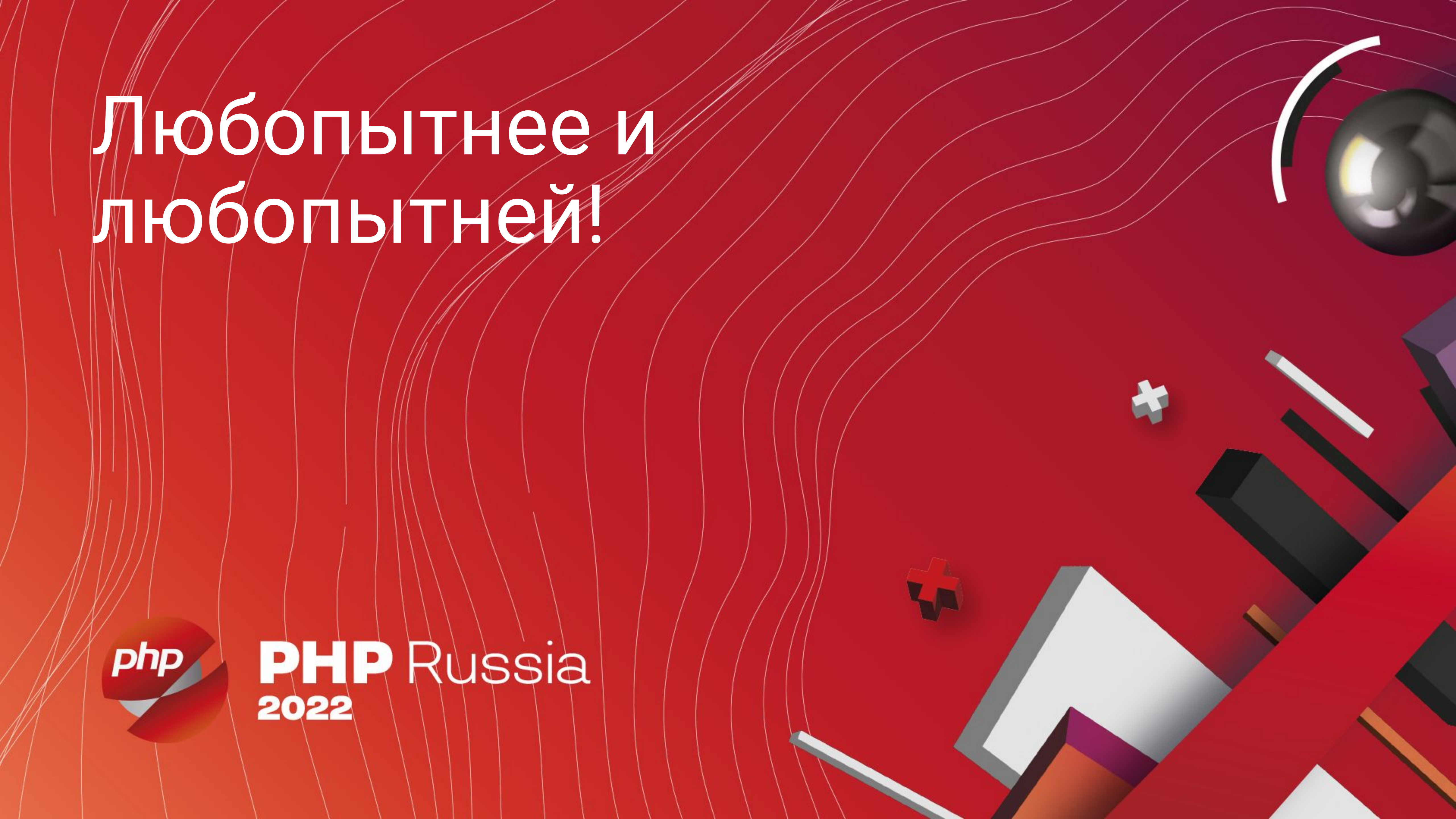
    public function getResult(): mixed
    {
        return $this->result;
    }
}
```



ReactPHP






Любопытнее и любопытней!
php
PHP Russia
2022

Promise
Resolve
Promise
Resolve
Start
Task A
Waiting...
Task A
Result
Time
Start
Task B
Waiting...
Task B
Result
External Request
Reference
External Request
Reference
Check by Reference
Result
Check by Reference
Result
Data Store Client


php
PHP Russia
2022

Promise resolve

Result A

Data store request

Promise B

Business logic

Task start

Task B

Tasks Pool

Task A

Task start

Business logic

Data store request

Promise A

Promise resolve

Result B



```php
<?php

class EventLoop
{
    public function __construct(
        private iterable $stasksPool
    ) {}

    public function addTask(callable $task): void
    {
        $this->tasksPool[] = $task;
    }

    public function start(): self
    {
        $this->tick();

        return $this->start();
    }
}
```



ReactPHP

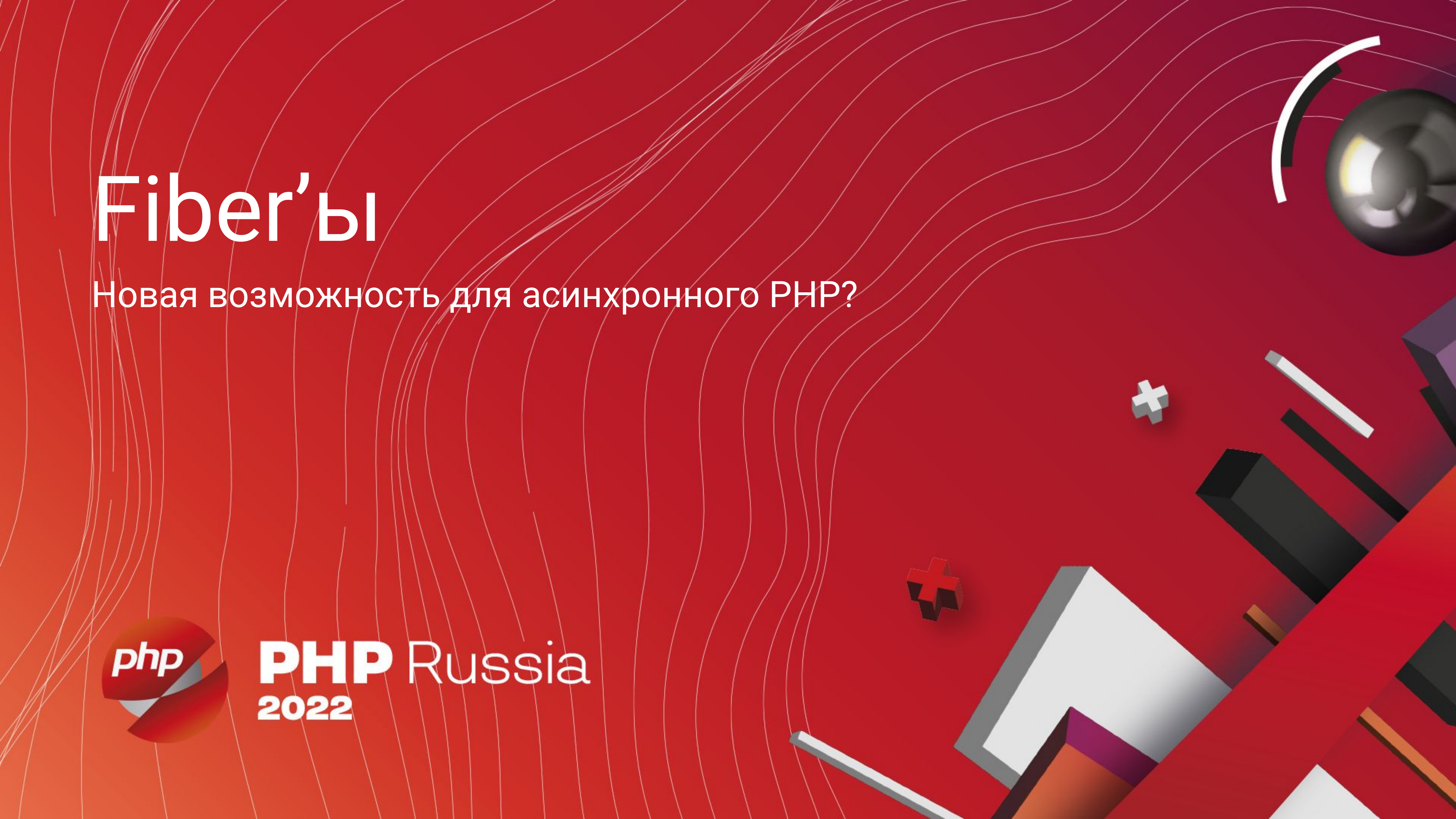






Скажите, пожалуйста, куда мне отсюда идти?




Fiber'ы
Новая возможность для асинхронного PHP?
php
PHP Russia
2022

Что это такое?

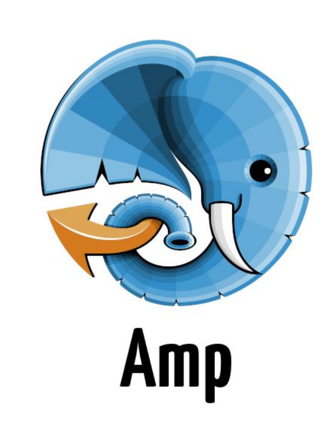
Fiber'ы — это примитивы для реализации облегченного кооперативного параллелизма [*в Ruby*]. По сути, они являются средством создания блоков кода, которые можно приостанавливать и возобновлять подобно потокам. Основное отличие состоит в том, что они никогда не вытесняются и планирование должно выполняться программистом, а не виртуальной машиной.




Принесли ли Fiber’ы асинхронность?
php
PHP Russia
2022


php
PHP Russia
2022


REACTPHP

SWOOLE

Amp

php
PHP Russia
2022

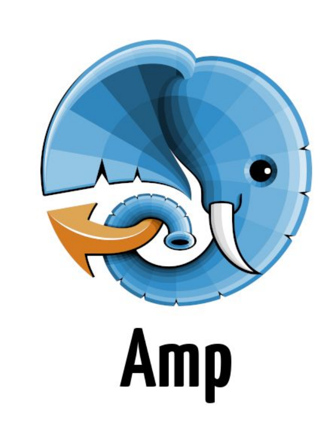

Нужны ли нам Fiber'ы?
php
PHP Russia
2022

~~Fiber’ы~~

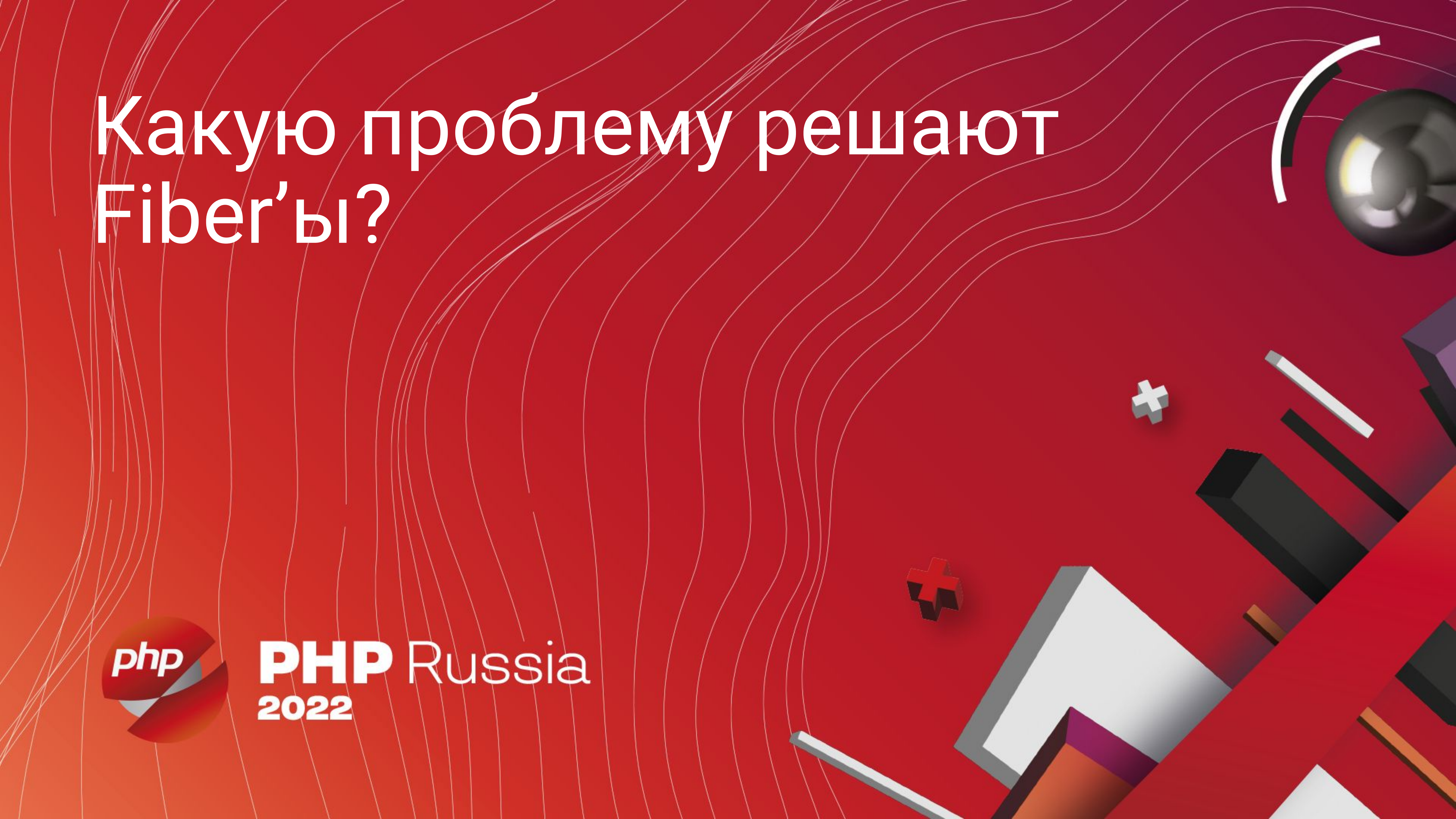



REACTPHP


SWOOLE


Amp


php
PHP Russia
2022


Какую проблему решают Fiber'ы?
php
PHP Russia
2022

Fiber’ы отвечают на вопрос:





Синхронный

```php
<?php

class UserRepository
{
    private string $base = 'http://example.com/user/';

    public function checkUser(int $id): bool
    {
        $response = fetch($this->base . $id);

        return 200 === $response->getStatusCode();
    }
}

$ok = (new UserRepository)->checkUser(42);
if ($ok) {
    echo 'User exists!';
}
```



Promises

```php
<?php

class UserRepository
{
    private string $base = 'http://example.com/user/';

    public function checkUser(int $id): PromiseInterface
    {
        return fetch($this->base . $id)->then(function(ResponseInterface $response): bool {
            return 200 === $response->getStatusCode();
        });
    }
}

(new UserRepository)->checkUser(42)->then(function(bool $ok): void {
    if ($ok) {
        echo 'User exists!';
    }
});
```



Coroutines

```php
<?php

class UserRepository
{
    private string $base = 'http://example.com/user/';

    public function checkUser(int $id): PromiseInterface
    {
        return async(function() use ($id) {
            $response = yield fetch($this->base . $id);

            return 200 === $response->getStatusCode();
        });
    }
}

(new UserRepository)->checkUser(42)->then(function(bool $ok): void {
    if ($ok) {
        echo 'User exists!';
    }
});
```



Fiber’ы

```php
<?php

class UserRepository
{
    private string $base = 'http://example.com/user/';

    public function checkUser(int $id): bool
    {
        $response = fetch($this->base . $id);

        return 200 === $response->getStatusCode();
    }
}

$ok = (new UserRepository)->checkUser(42);
if ($ok) {
    echo 'User exists!';
}
```




Как выглядит параллелизм
в реальных приложениях?
php
PHP Russia
2022

Синхронный

```php
<?php

class UserRepository
{
    private string $base1 = 'http://example.com/user/';
    private string $base2 = 'http://api.example.org/user/';

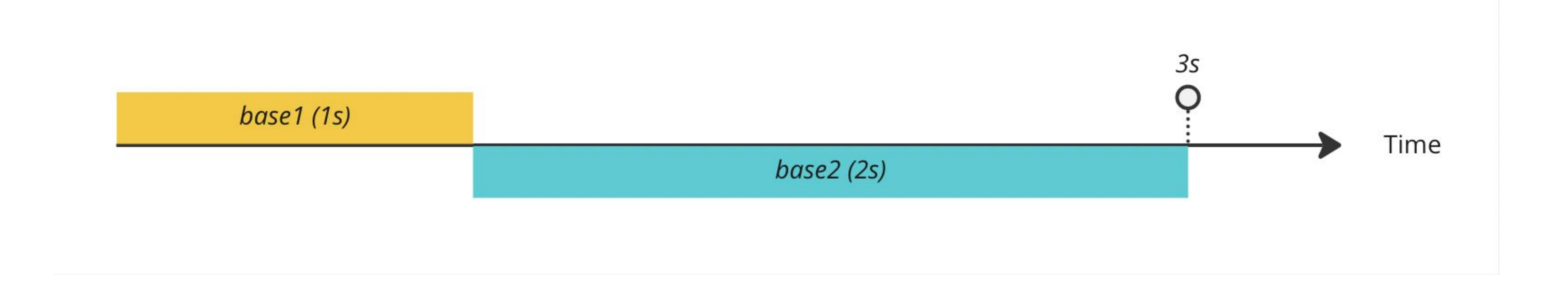
    public function checkUser(int $id): bool
    {
        $response1 = fetch($this->base1 . $id);
        $response2 = fetch($this->base2 . $id);

        return 200 === $response1->getStatusCode() && 200 === $response2->getStatusCode();
    }
}

$ok = (new UserRepository)->checkUser(42);
if ($ok) {
    echo 'User exists!';
}
```



Синхронный





Promises

```php
<?php

class UserRepository
{
    private string $base1 = 'http://example.com/user/';
    private string $base2 = 'http://api.example.org/user/';

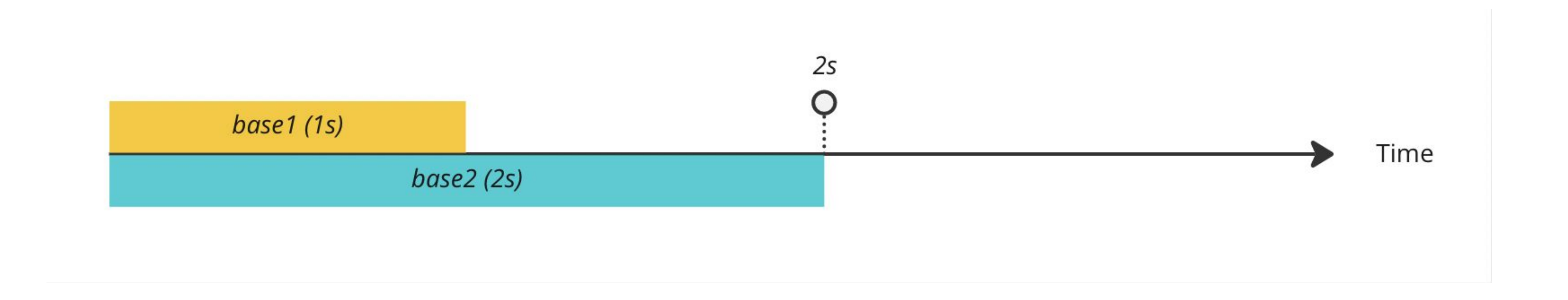
    public function checkUser(int $id): PromiseInterface
    {
        $promise1 = fetch($this->base1 . $id);
        $promise2 = fetch($this->base2 . $id);

        return all([$promise1, $promise2])->then(function (array $responses) {
            return 200 === $responses[0]->getStatusCode() && 200 === $responses[1]->getStatusCode();
        });
    }
}

(new UserRepository)->checkUser(42)->then(function(bool $ok): void {
    if ($ok) {
        echo 'User exists!';
    }
});
```



Promises





Coroutines

```php
<?php

class UserRepository
{
    private string $base1 = 'http://example.com/user/';
    private string $base2 = 'http://api.example.org/user/';

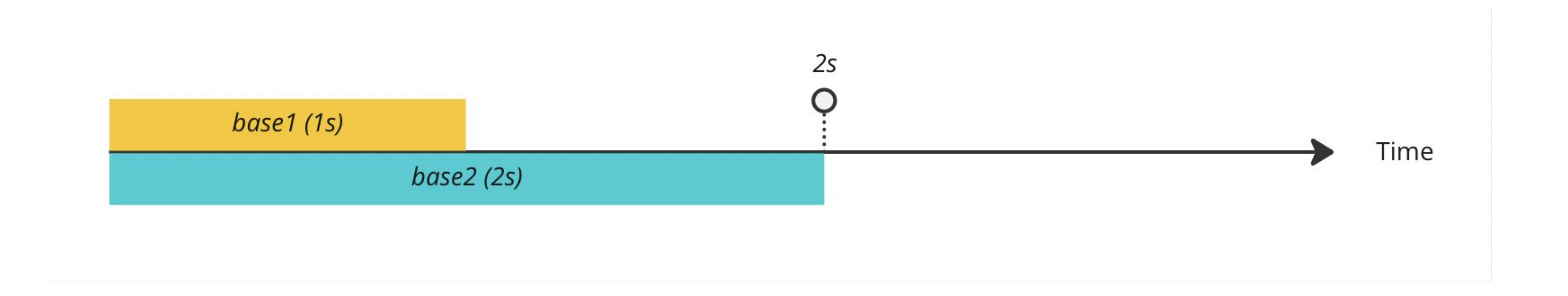
    public function checkUser(int $id): PromiseInterface
    {
        return async(function () use ($id) {
            $promise1 = fetch($this->base1 . $id);
            $promise2 = fetch($this->base2 . $id);

            $responses = yield all([$promise1, $promise2]);
            return 200 === $responses[0]->getStatusCode() && 200 === $responses[1]->getStatusCode();
        });
    }
}

(new UserRepository)->checkUser(42)->then(function(bool $ok): void {
    if ($ok) {
        echo 'User exists!';
    }
});
```



Coroutines





Fiber'ы

```php
<?php

class UserRepository
{
    private string $base1 = 'http://example.com/user/';
    private string $base2 = 'http://api.example.org/user/';

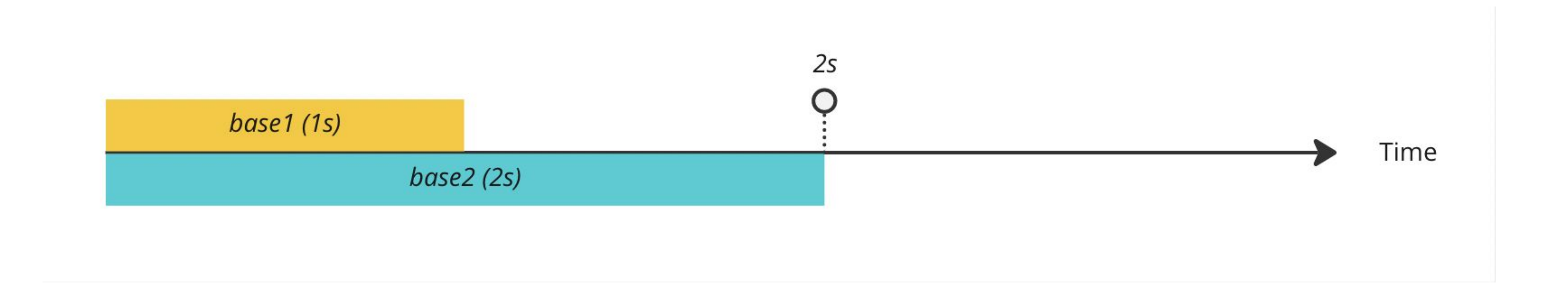
    public function checkUser(int $id): bool
    {
        $promise1 = async(function () use ($id) {
            return fetch($this->base1 . $id);
        });
        $promise2 = async(function () use ($id) {
            return fetch($this->base2 . $id);
        });

        $responses = await(all([$promise1, $promise2]));
        return 200 === $responses[0]->getStatusCode() && 200 === $responses[1]->getStatusCode();
    }
}

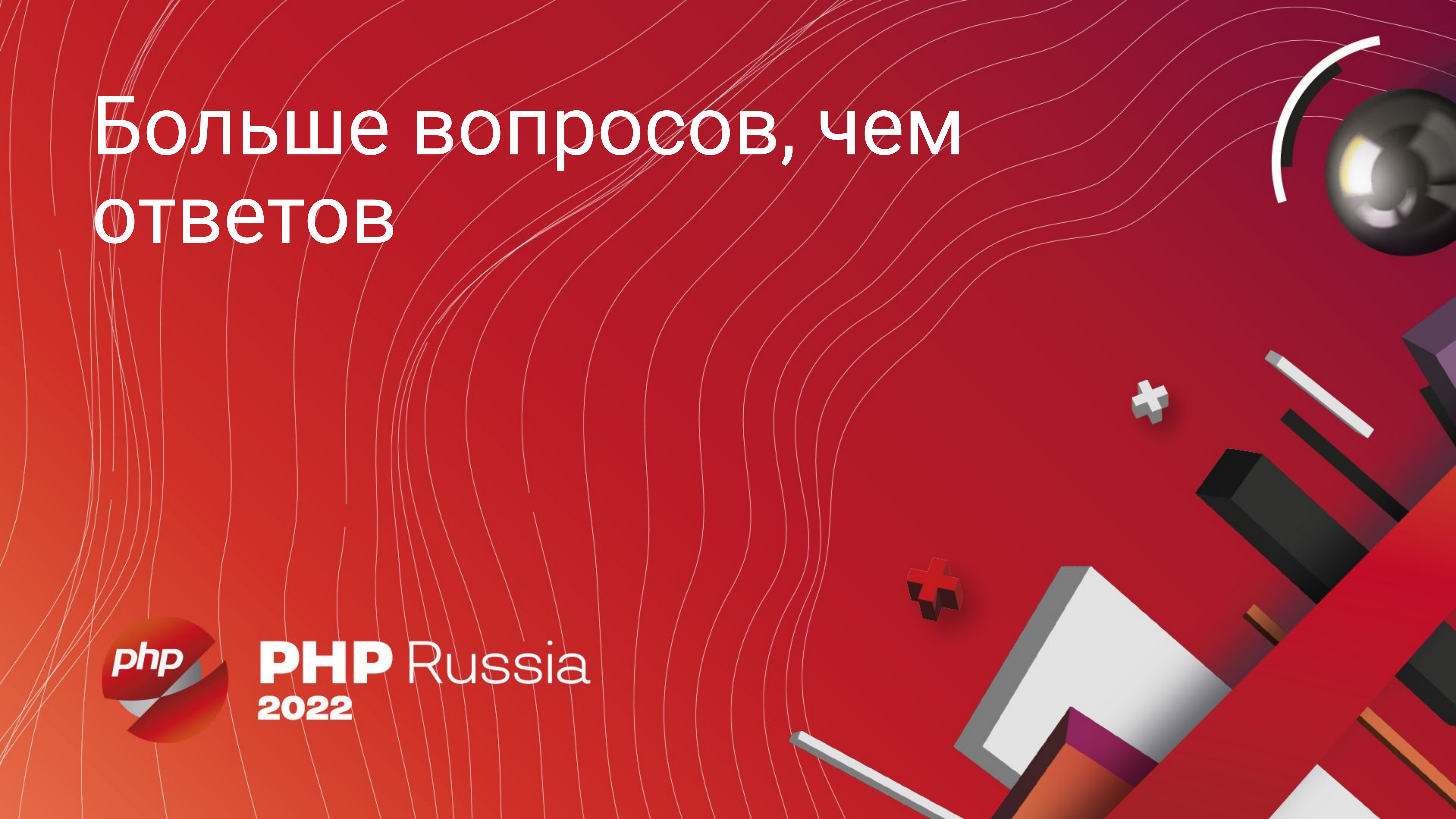
$ok = (new UserRepository)->checkUser(42);
if ($ok) {
    echo 'User exists!';
}
```



Fiber'ы






Больше вопросов, чем ответов
php
PHP Russia
2022

Вопросы и ответы

- Что это значит для будущего Promises?



Вопросы и ответы

- Что это значит для будущего Promises?
- Что это значит для будущего Coroutines?



Вопросы и ответы

- Что это значит для будущего Promises?
- Что это значит для будущего Coroutines?
- Как насчет async/await?



Вопросы и ответы

- Что это значит для будущего Promises?
- Что это значит для будущего Coroutines?
- Как насчет async/await?
- Должен ли PHP иметь Fiber’ы?



Если бы это было так, это бы ещё ничего. Если бы, конечно, оно так и было. Но так как это не так, так оно и не этак. Такова логика вещей.





@dslonyara

d@mamontov.tech



Обратная связь
и комментарии по
докладу по ссылке



